Architectures Toward Reusable Science Data Systems

John.f.Moses@nasa.gov

Science Data Systems Branch, NASA Goddard Space Flight Center, Greenbelt, MD 20771

Science Data Systems (SDS) comprise an important class of data processing systems that support product generation from remote sensors and in-situ observations. These systems enable research into new science data products, replication of experiments and verification of results. NASA has been building systems for satellite data processing since the first Earth observing satellites launched and is continuing development of systems to support NASA science research and NOAA's Earth observing satellite operations. The basic data processing workflows and scenarios continue to be valid for remote sensor observations research as well as for the complex multi-instrument operational satellite data systems being built today.

Satellite Data System Enterprise Architectures

- Business process description focus on:
  - Dynamic interaction of stakeholders; roles & interfaces
  - Flow of information between the enterprise elements
- Business model can drive design; identifies stakeholders, systems and data; examples include:
  - NASA EOSDIS science discipline-specific facilities such as Science Investigator-led Processing systems (SIPS) and Distributed Active Archive Centers (DAACs);
  - Joint Polar Satellite System (JPSS) has mission partner facilities/systems; e.g., NOAA NESDIS STAR, ESPC, FNMOC, CLASS, NASA SDS
- Manages interfaces; enables system design independence

Examining Satellite Science Data System Architectures

- Look for generalize reoccurring structures and properties: e.g. file transfer, job control, algorithm input data and run configuration
- Characterize features most important to developers and operators: e.g., functional, performance, Maintainability
- Test methods to scale/extrapolate scenario

Software Architecture Views for Re-use

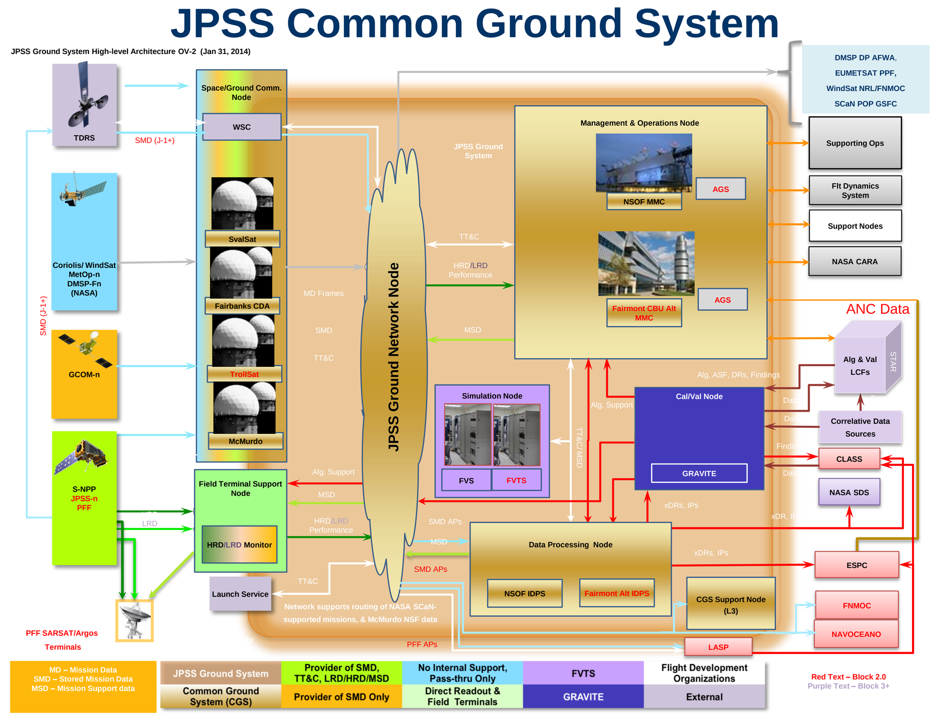
- Establish a design hierarchy and process, structure of elements, properties and relationships, abstractions for managing complexity
- Partition the system into software elements (components) with responsibilities and interaction (interface) rules, hierarchical, recursive with focus on functionality
- Look for conceptual integrity: a small number of simple interaction patterns. System functions such as ingest, product generation and distribution need to be configured and perform consistently with scalability
- Re-use infrastructure, framework, data models

Architect's Application

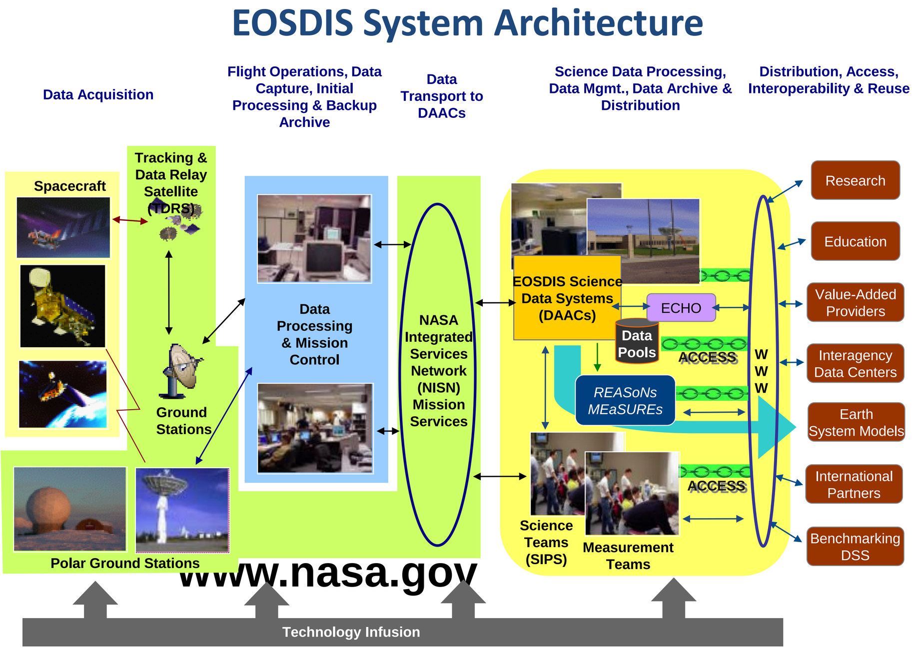
Use Aura OMI ozone instrument science data processing scenario to serve as model of priority functions for examining solution attributes

- Science algorithm scenario allows partitioning into sets of the most basic or general functions and interactions
- Frameworks concept prescribes the design methodology
  - Two supporting middleware packages emerge as popular frameworks
- Abstract views are used to identify components with common structures and priority attributes

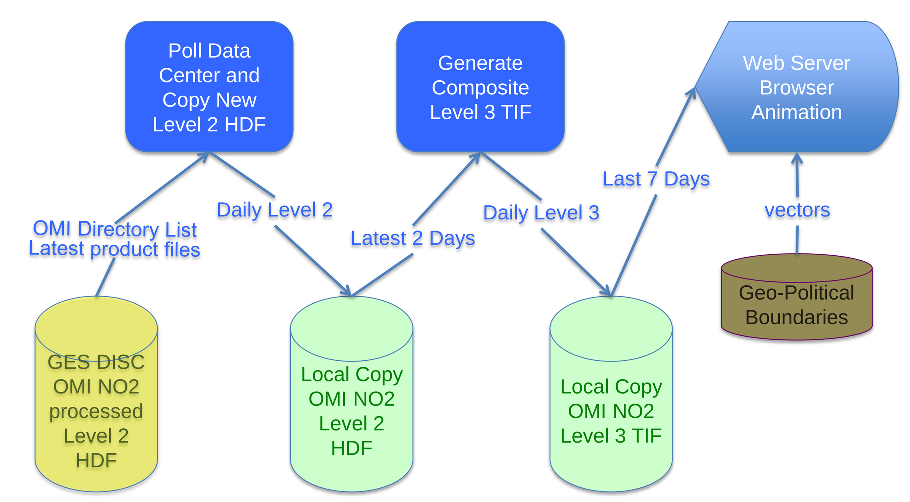
JPSS Common Ground System



EOSDIS System Architecture



Simplistic Satellite Science Data System Use Case Scenario



Aura OMI instrument observations of NO2 (Tropospheric NO_2) in Level 2 (by orbit) format are acquired from the GES DISC and used to make multi-day Level 3 global grid for visual display.

Major Functions for Satellite Science Data System:

Acquire calibrated and geo-location instrument observations covering their operating life

- File transfer protocols and methods
  - Configure for FTP, SFTP, or HTTP file transfers
    - User provides information about the type and internet location of instrument observation data
    - Data subscription with data center source protocols
    - Copy observation time/location-based data files to local directory
- Extract metadata for downstream process control'
  - Support common file formats with standard metadata content: e.g., HDF, NetCDF, ISO 19115
  - Provide key content: data observation/model time, spatial resolution and coverage extent
  - Source identification: file name, headers internal to the file and/or separate configuration file

Generate higher level synoptic-based products

- The algorithm assimilates (e.g., composites) multiple observation times into a representative time period
- Integrates other external sources of observations, model or reference geophysical parameters
- Configure run criteria and data format for algorithms
  - Identify all observations and static inputs
- Run algorithm process scripts and executables when all input data is available
- Store results locally for distribution, downstream analysis, visualization

GRAVITE Software Architecture

JPSS node; Government Resource for Algorithm Verification Integration and Test Environment (GRAVITE) data system built using Apache Object Oriented Data Transfer (OODT) framework

1. JAVA in Linux server environment
2. Process steps use components from OODT
3. Communicate via XML Remote Procedural Calls

Instrument data systems employing OODT components:

- Seawinds/QuickSCAT science data processing
- SMAP: soil moisture science data system (JPL)
- Orbiting Carbon Observatory-2: operations pipeline (JPL)
- SNPP Sounder Product Evaluation & Test Element (PEATE)

GES DISC Software Architecture

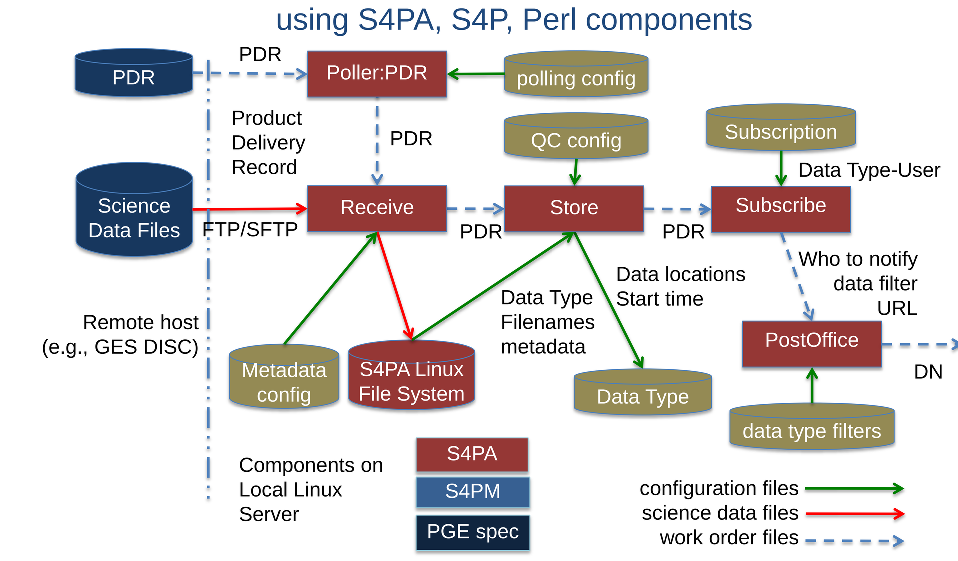
EOSDIS: Goddard Earth Sciences Data and Information Services Center (GES DISC) Science Data System built using Simple Scalable Script-based Science Processor (S4P)

1. Perl script, S4P Archive (S4PA), S4P Missions (S4PM)
2. Process steps are organized in directory structures
3. Station daemon and configuration file provide building blocks: Polls local directory for work order files, looks up commands for type of work, changes to temporary subdirectory, forks child process to execute the job, creates and writes output work order to downstream station

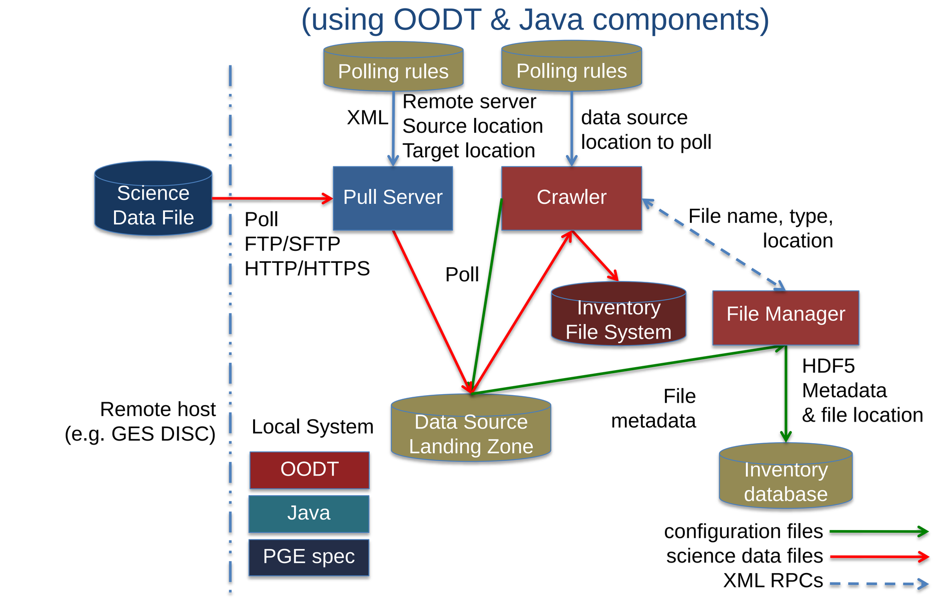
Instrument data systems employing S4P and Perl-based framework components: (sample)

- TRMM science data system: (GES DISC)
- AQUA AIRS and AURA MLS, OMI: (GES DISC & OMI SIPS)
- TERRA ASTER: ASTER on-demand system (LP DAAC)
- TERRA MISR S4PM: (LARC ASDC)
- CALIPSO, FlashFLux S4PM (LARC ASDC)

Scenario Data Transfer using S4PA, S4P, Perl components



scenario data transfer process (using OODT & Java components)



Scenario Data Transfer using S4PA, S4P, Perl components

Poll PDR:

- Periodically looks in remote subscription PDR directory, pulls PDR files and sends them to Receive Data.
- Configuration file contains parameters for polling: e.g., remote host/directory, local directory for new PDRs, local file of accepted PDRs, polling protocol, format

Receive Data:

- Uses science data filename from PDR to create directory for the science data file
- Extracts metadata for data type, converts to XML
- Allocates local directory using PDR filename, download data file named in the PDR

Store Data

- Extracts metadata, stores data type records, obs time
- Looks in configuration for compression, quality check
- Creates and stores sym links to downloaded files
- Writes a subscription PDR containing sym links

Subscribe

- Reads the PDR file and extracts data type
- Configuration gives who to notify; data filters; URL
- Prepares PDR and sends to PostOffice for ftp or email

PostOffice

- Uses PDR to extract type and file metadata (XML)
- Configuration data type provides metadata filters
- Creates Delivery Notification (DN)

Scenario Data Transfer Process (using OODT & JAVA)

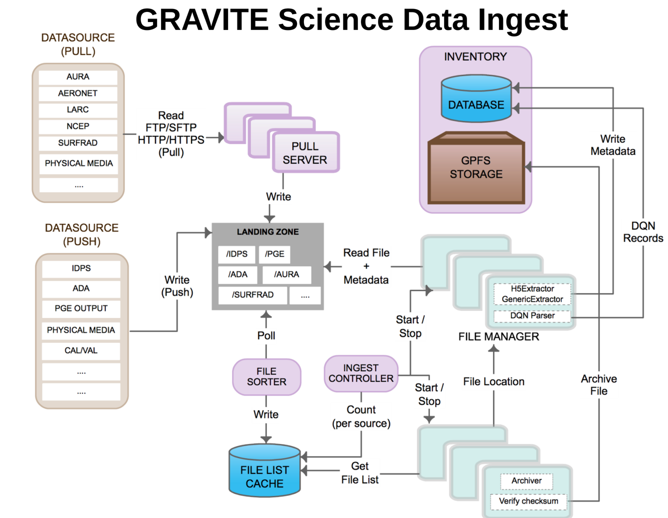
Pull Server

- Periodically checks in remote host location for new data files; transfers new files to source landing zone
- Configuration file contains polling parameters: e.g., remote host directory, source landing zone directory

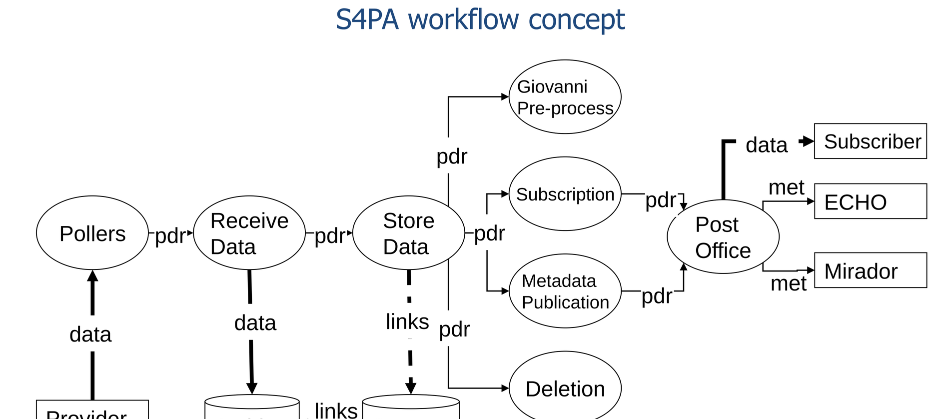
Crawler instances monitor data-source subdirectories for new files

- Verifies checksum; unique product identifier; and sends data type and file location to File Manager
- After successful database insert, moves file from landing zone to inventory
- File Manager receives file location, data type
- Extracts HDF5 and other metadata and populates the database. Sends message to Crawler on successful insert.

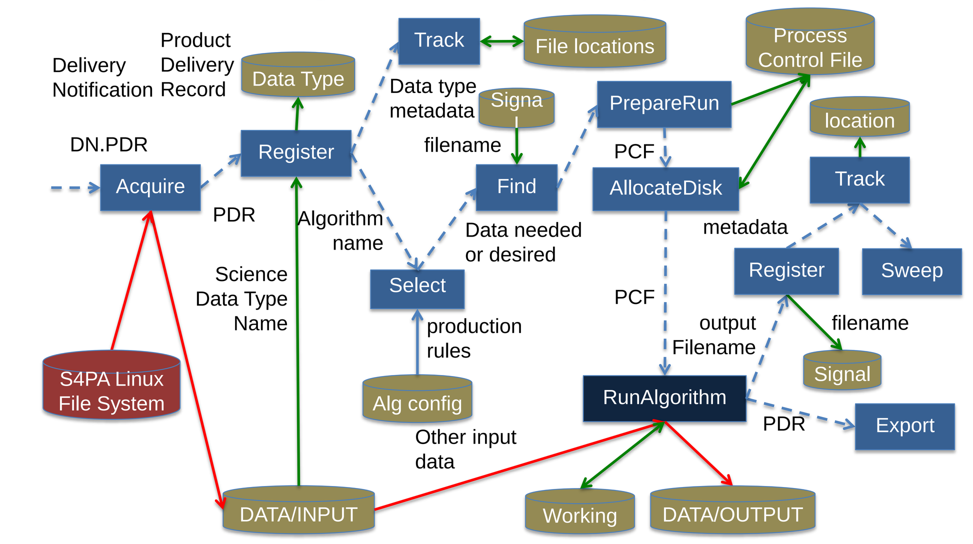
GRAVITE Science Data Ingest



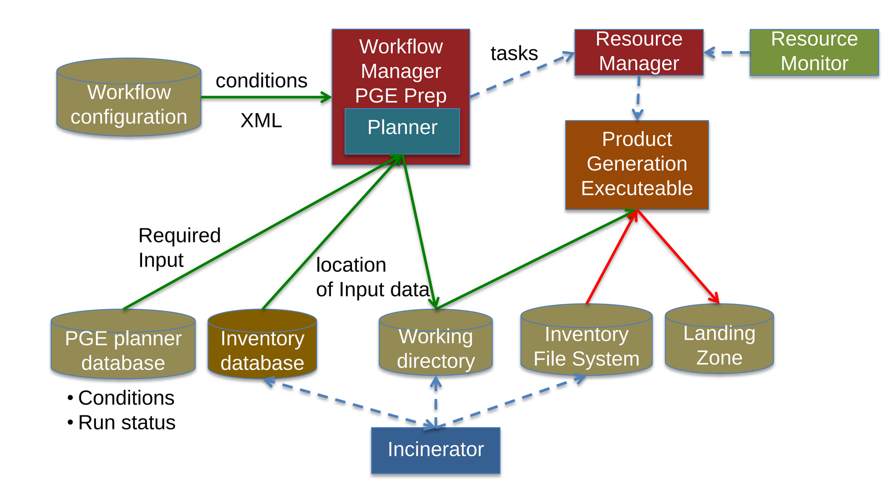
S4PA workflow concept



Scenario Workflow using S4PM, S4P, Perl components



Scenario Workflow (OODT & Java)



Scenario Workflow using S4PA, S4P, Perl

Acquire Data

- Reads DN.PDR for files to get
- Uses symlinks, or FTP get if remote
- Outputs PDR with data location

Register Data

- Uses data type to identify the algorithm name from configuration

Select Data

- Data type/time, production rules determine other required data

Track Data

- adds filename and finds expected algorithm uses in configuration

Find Data

- Locates the needed/desired inputs
- Outputs data found after timers expire

Prepare Run

- Creates a Process Control File using algorithm-specific template

Allocate Disk (S4PM)

- Allocates disk & adds directories to PCF

Run Algorithm (S4PM & code specific)

- Executes the named algorithm

Register Data (S4PM)

- Writes file name, metadata

Track Data (S4PM) – store type metadata and updates usage

Export (S4PM) – Writes PDR

Sweep (S4PM) - Deletes data file when use count drops to zero

Scenario Workflow using OODT & Java

Planner (Java)

- Verifies all input files in inventory
- Checks the inventory database for PGE inputs
- Tells the workflow Manager to create a working directory
- Updates PGE configuration files in the working directory

WorkFlow Manager (OODT)

- reads config of conditions & tasks
- Creates a workflow instance and processing thread
- Creates a working directory with symbolic links to the input files
- Send the executable tasks to the Resource Manager

Resource Manager (OODT)

- Resource Monitor determines state of resources on the servers
- Sends jobs to queue/scheduler when resources are available
- Batch Managers submit jobs to Resource Nodes on the servers

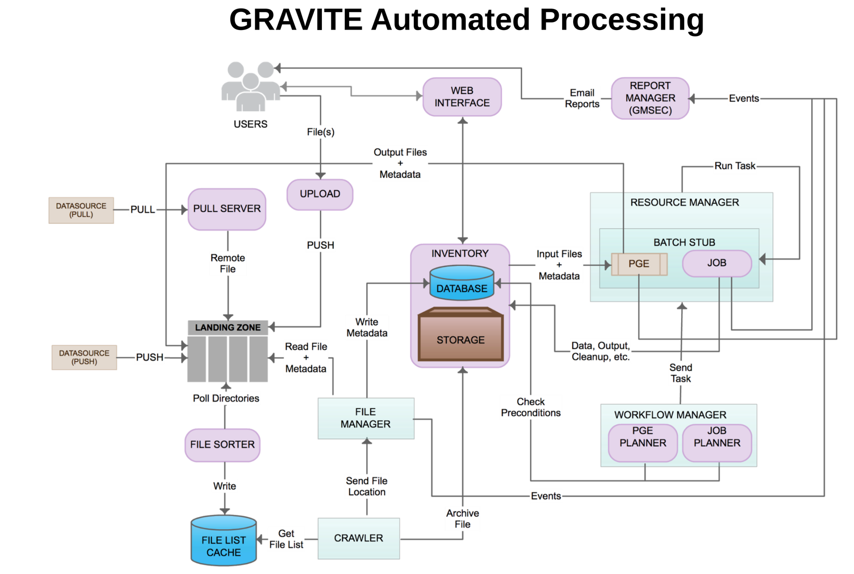
PGE (JAVA, PGE specific languages)

- Executes algorithms/commands
- Output moved into landing zone

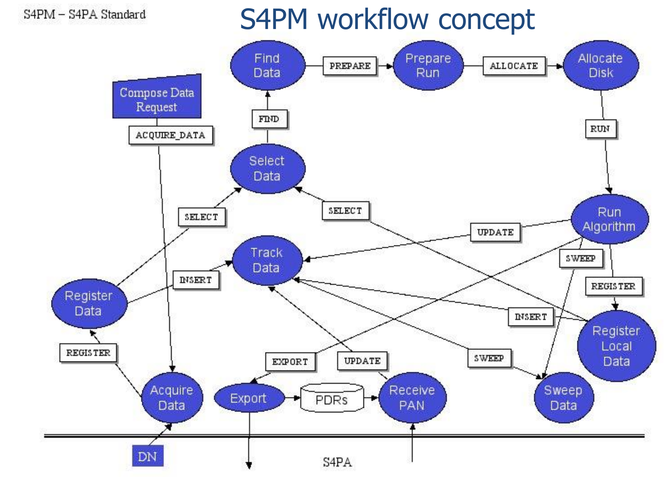
Incinerator (JAVA)

- Periodically searches and removes links and folders after time expires

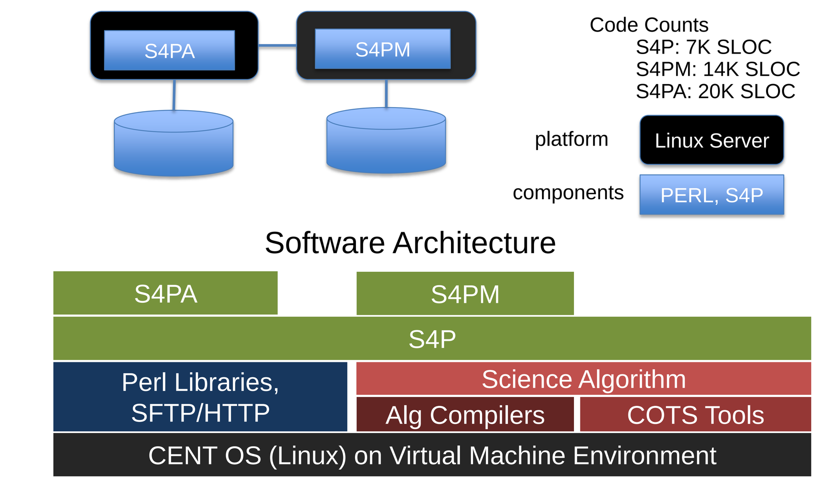
GRAVITE Automated Processing



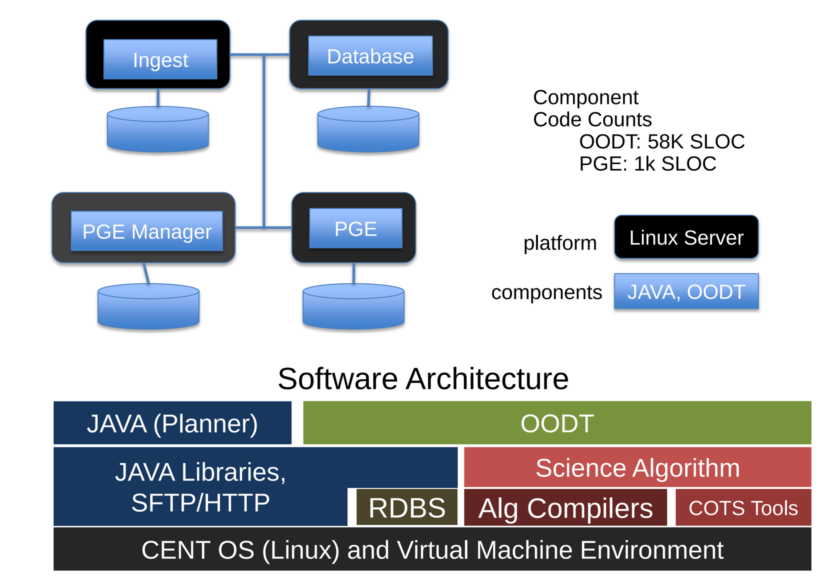
S4PM workflow concept



Scenario S4P, S4PA, S4PM Scenario Deployment View



Scenario Deployment View (OODT & JAVA)



Summary Highlights and Distinctions

Two middleware frameworks are used in many current satellite science data systems. They provide the major functions for supporting simple science data processing scenarios and offer practical reuse options at the component level.

- Data download and storage management
- Workflow management and algorithm application

They are composed of similar processing steps

- Science data transfer using standard directory polling and data protocols
- Workflow chain development for instrument data processing algorithms

Reuse is made possible through public software release and by availability of limited informal set of code examples, design artifacts and user guides.

Future Work

- Examine implementations to quantify latency and scalability factors.
- Understand complexity in installation, tuning and configuration management.
- Quantifying the significance of language skill requirement for Perl vs. Java.

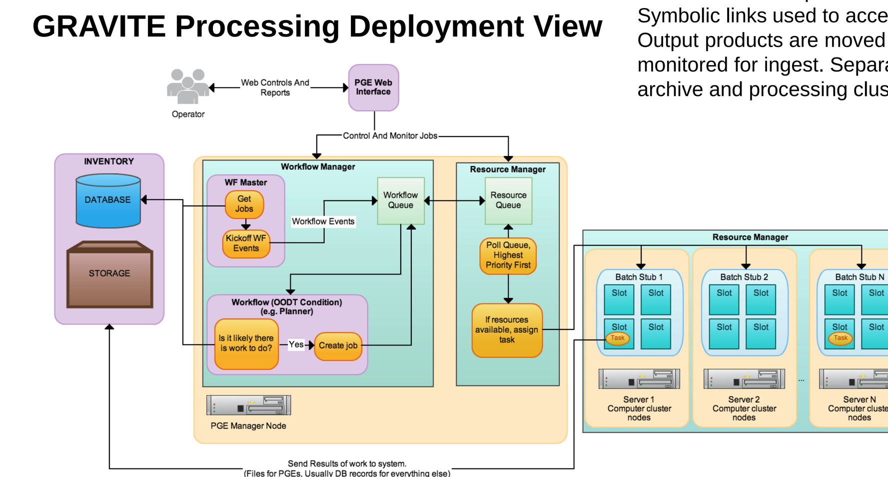
Perl, S4P, S4PA, S4PM

- S4P is a framework for S4PA and S4PM, where a standard station daemon polls for new work order files in local directory and maintains a queue.
- Scripts and configurations are added for S4PA and S4PM functions, includes handing addition popular protocols and metadata.
- Communicates among stations uses the file system and includes several conventional protocols.
- S4PA functions use station configurations to control data transfer by polling remote host for available data location, then constructing request to transfer the remote data. A directory is created in local file system from filename, and symbolic links for access.
- S4PM includes major functions in station components, stations look for and prepare inputs, run algorithm on dedicated resource. Load balance via static configuration parameters.
- Creates S4PM location for output files; links or moves them to S4PA. Archive is separate from algorithm processing platform.

OODT, JAVA

- OODT functions are in Java components grouped into data ingest and workflow management.
- Java methods and configurations are added to support data type ingest and algorithm execution planning functions.
- Communicates among components using XML RPCs (XML encoding, HTTP)
- Data transfer controlled through two polling components, one polls for files in remote subscription directory and transfers them to local directory, second polls for files in local directory and moves them to an inventory file system. Utilization is maximized and delays are minimized through tuning timers and other parameters.
- Functionality added to interface and manage science data configurations and data inventory, preparing input data for running algorithms on dedicated resources.
- Job queues and resource queues are used to control and run algorithm in working directories on computer cluster nodes. Symbolic links used to access science data. Output products are moved to file system monitored for ingest. Separate platforms for archive and processing cluster.

GRAVITE Processing Deployment View



Aura Ozone (OMI) instrument data processing and archive at GES DISC